Automotive Immersion Day Labs

This hands-on lab will guide you through a series of steps to showcase how AWS services fit into the context of Automotive. The labs will help you to understand device connectivity, data generation, real-time notification, and the analysis of the generated data.

# Prerequisites

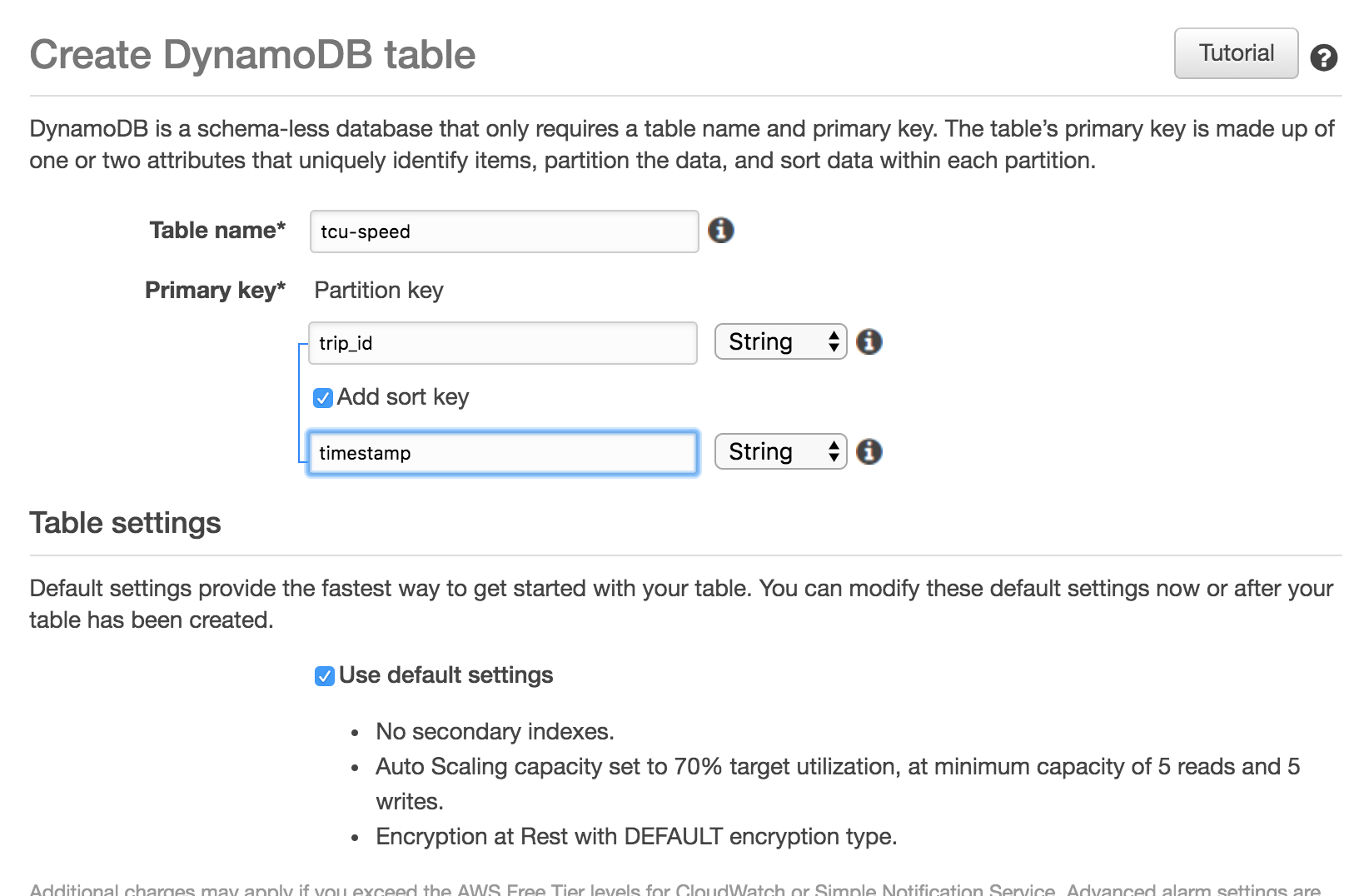
### Lab1 - Connected Vehicle Immersion Day: Vehicle to tcu Secure Connectivity

# Lab 2: Store Device Generated Data

In this lab, we will configure a rule in IoT to send data generated by our **tcu** (name of your device) thing to a DynamoDB table.

## Step 1 – Create DynamoDB Table

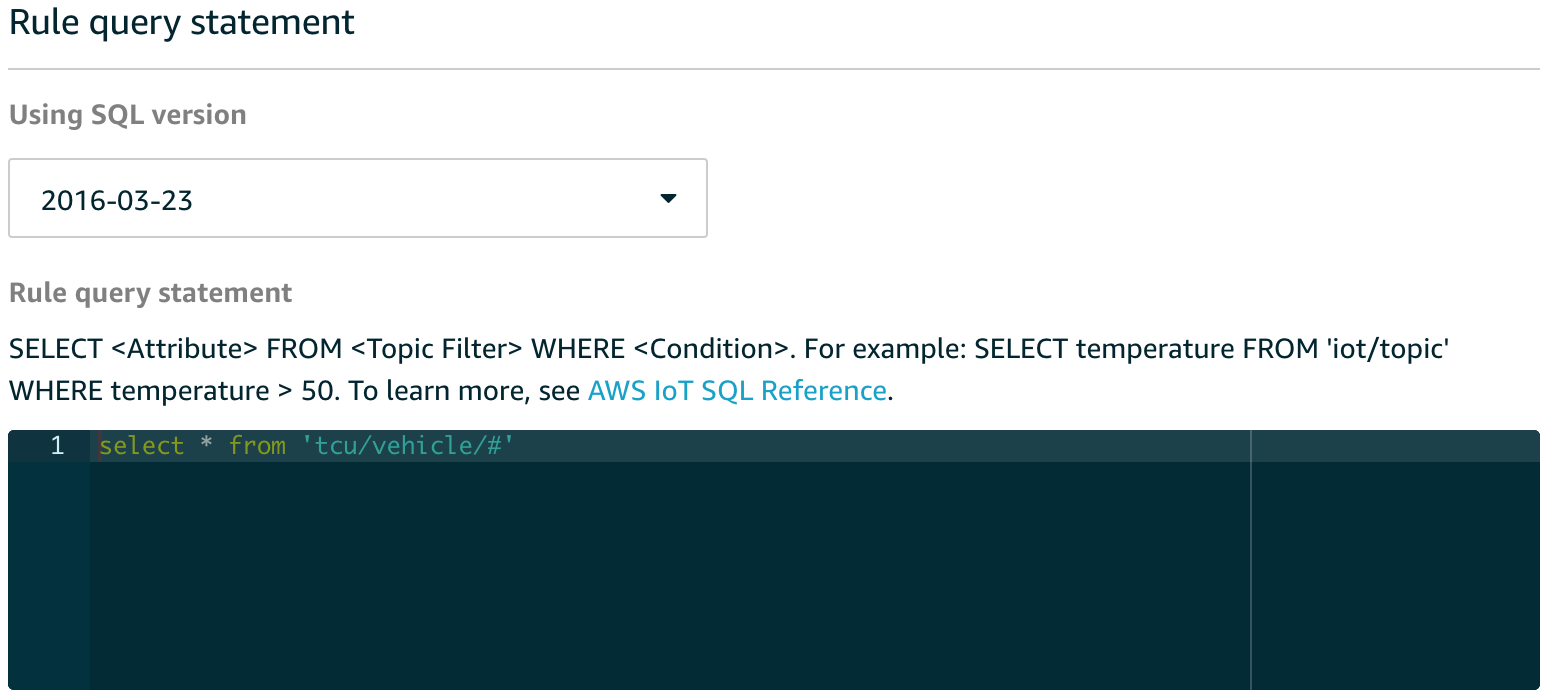
Create a DynamoDB table called **tcu-speed** with Partition **trip\_id** and Sort Key **timestamp.** Datatypewould be **String** for both.



## Step 2 – Setup IoT Rule

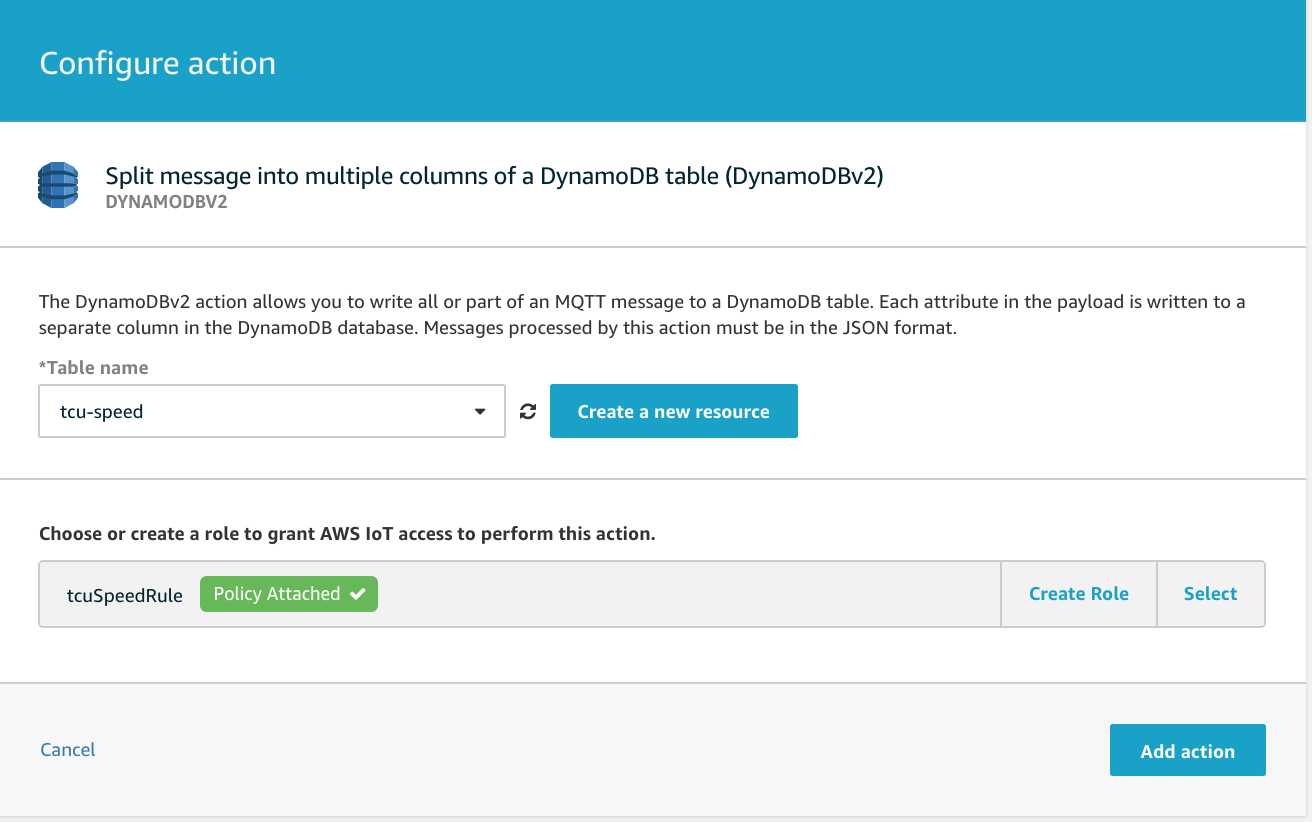
Add a rule and action to our AWS IoT setup that will listen to the tcu/vehicle topic (all VINs) and pass messages to DynamoDB.

1. Open the AWS IoT dashboard and click **Act** and choose **Create**.
2. Create a rule with the following values:
3. Name: *tcuSpeedRule*
4. Description: “Rule to pass messages to DynamoDB table tcu-speed”
5. Rule query statement: SELECT \* FROM 'tcu/vehicle/#'



1. Add Action: **Split message into multiple columns of a DynamoDB table (DynamoDBv2)** and select **Configure Action**
2. Choose the **tcu-speed** table that you created in Step 1
3. Click on **Create** **Role** (e.g.; tcuSpeedRule) to grant AWS IoT permissions to perform the action (write to the tcu-speed table)
4. Add Action

**Configure action** page as shown below:



1. Click on **Create Rule**.

## Step 3 – Generate Data and Verify Result

Run the **tcu.py** script in the Cloud9 IDE. You should be able to see data in the **tcu-speed** table. One entry is created for each timestamp.

# Lab 3: Send Speed Alert in Real-Time

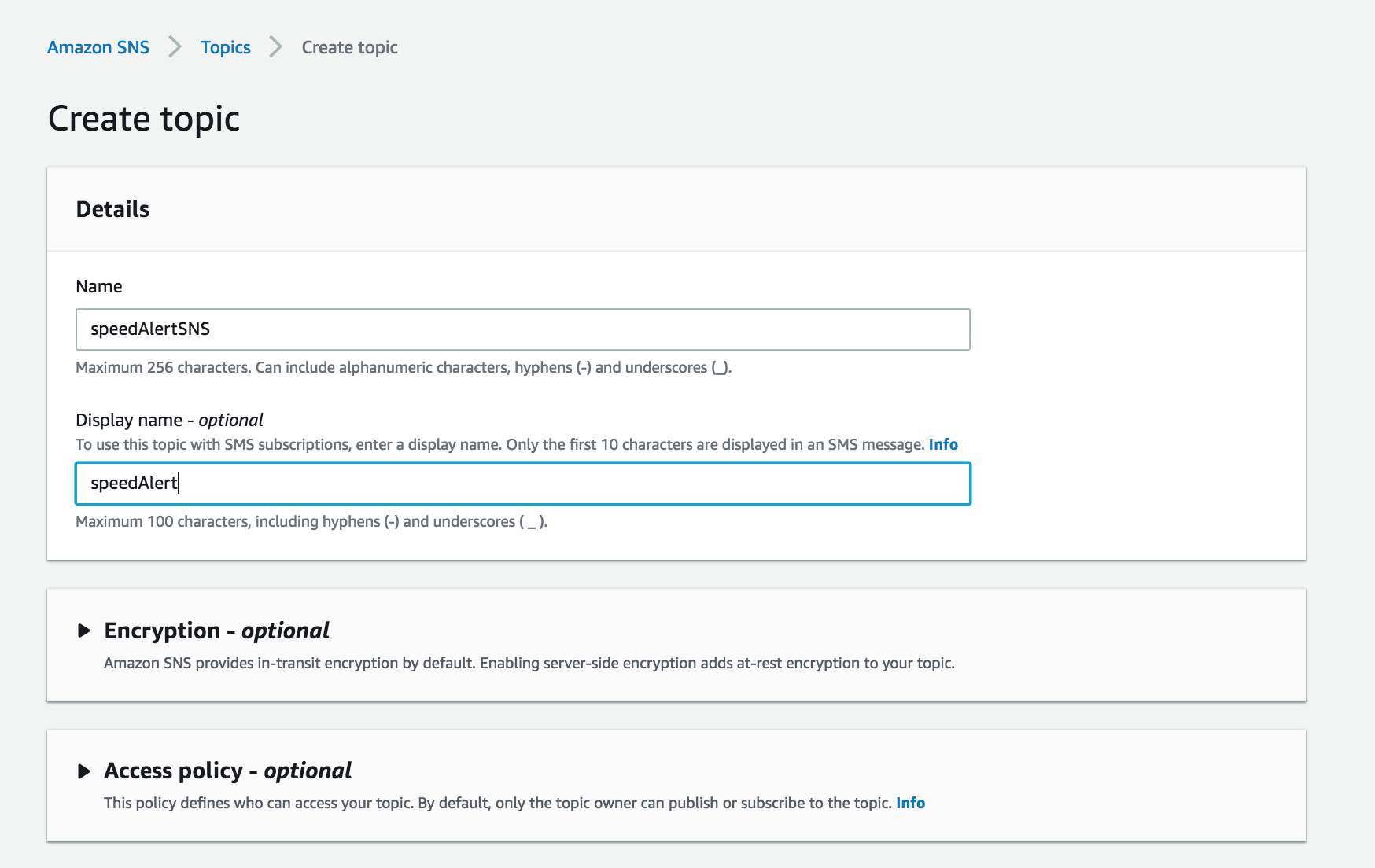
The Amazon Simple Notification Service (Amazon SNS) is the key to good architecture and helps build a pluggable pattern that makes it easy to extend your application in the future. This is critical for IoT system design since, in many cases, you’ll want not only guaranteed message delivery but also guaranteed actions on those messages.

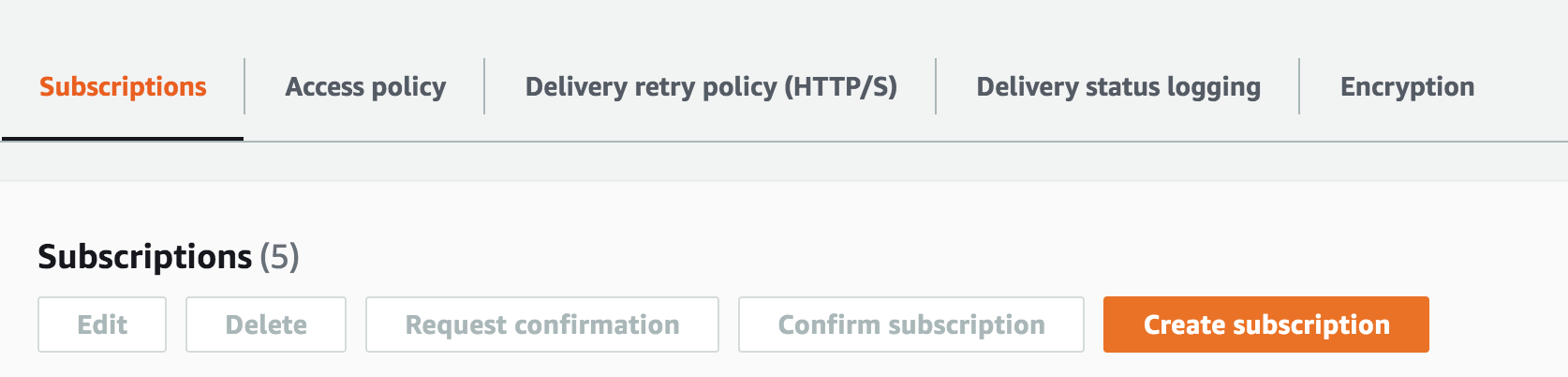
For this lab we’ll setup an SNS topic, route messages from AWS IoT to SNS, and then show how to use Amazon Simple Queue Service (Amazon SQS) to recover messages from a failure.

## Step 1 - Create a New SNS topic

Let’s setup a new SNS topic to notify us when the tcu’s speed hits a certain threshold.

1. Log into the AWS Console.
2. Open the SNS dashboard and create a new topic. We’ll call this topic **speedAlertSNS**.
3. Make a note of the ARN for this new topic.

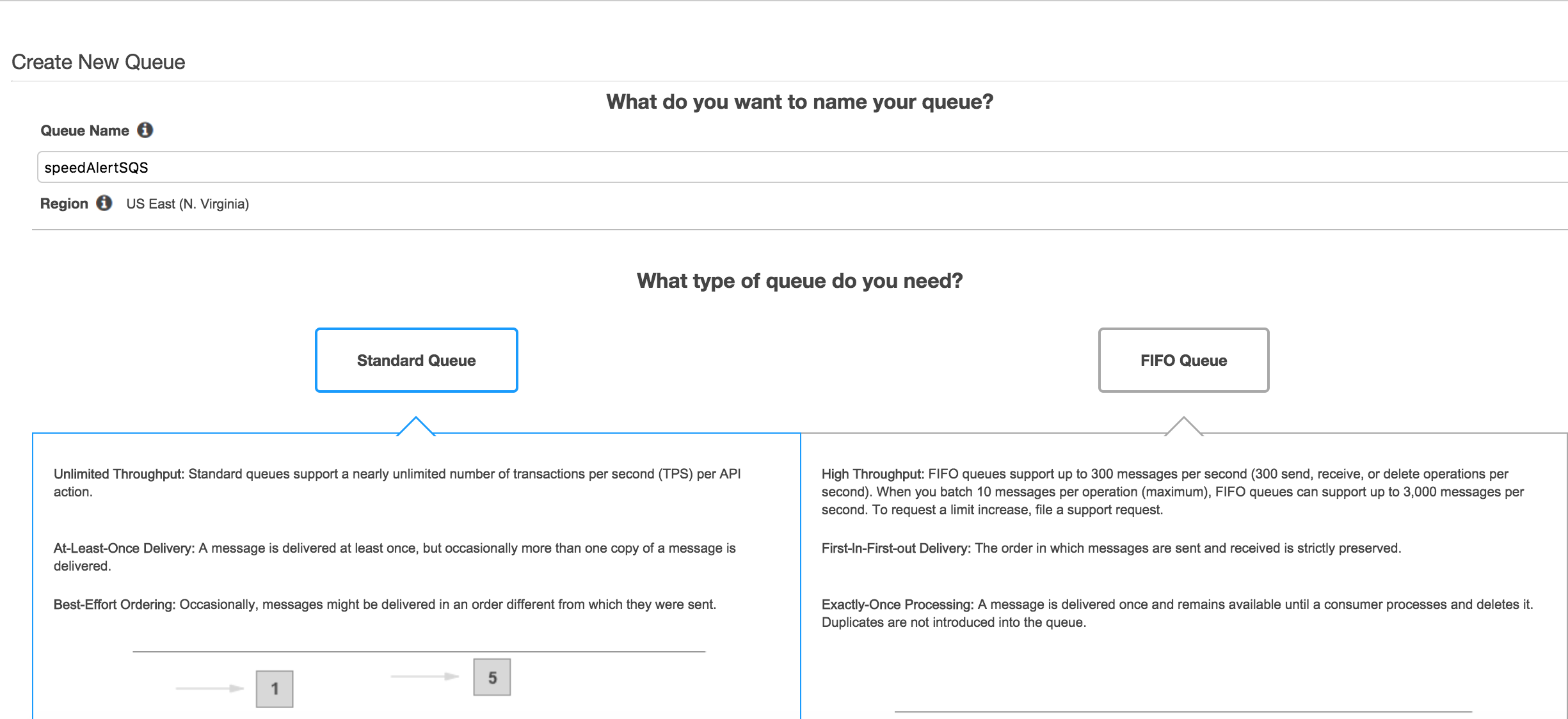


1. After the topic is created, select it, and click **Create Subscription**.   
     
   
2. On the **Create subscription** page, select the Protocol **SMS** and Endpoint **Enter your mobile number,** and click **Create Subscription**. Note: For India numbers SMS may not work. In that case instead create Email subscription and provide your email ids where you want to receive the alert.

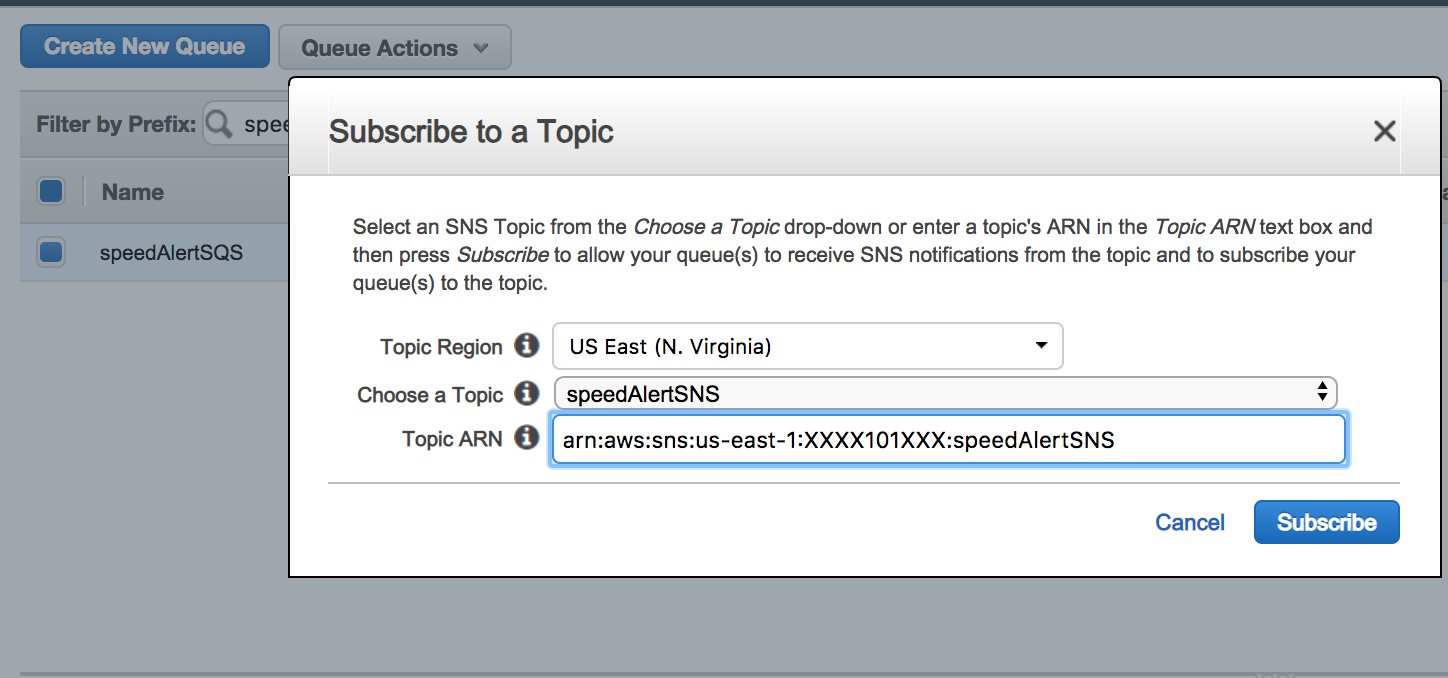
## Step 2 - Creating a new SQS queue

Let’s setup a new SQS queue so that we could recover messages in the event of a failure/misconfiguration of the SNS service or transport. Any messages or alerts coming into our SNS topic will also be enqueued in the speedAlertSQS queue.

1. Open the SQS dashboard and create a new queue. We’ll call this queue **speedAlertSQS**
2. Select **Standard Queue.**
3. Click **Quick-Create Queue.**



1. Subscribe the queue to the SNS topic created earlier, **speedAlertSNS**. Right-click the queue and select **Subscribe Queue to SNS topic**.



Make a note of the ARN for this new queue

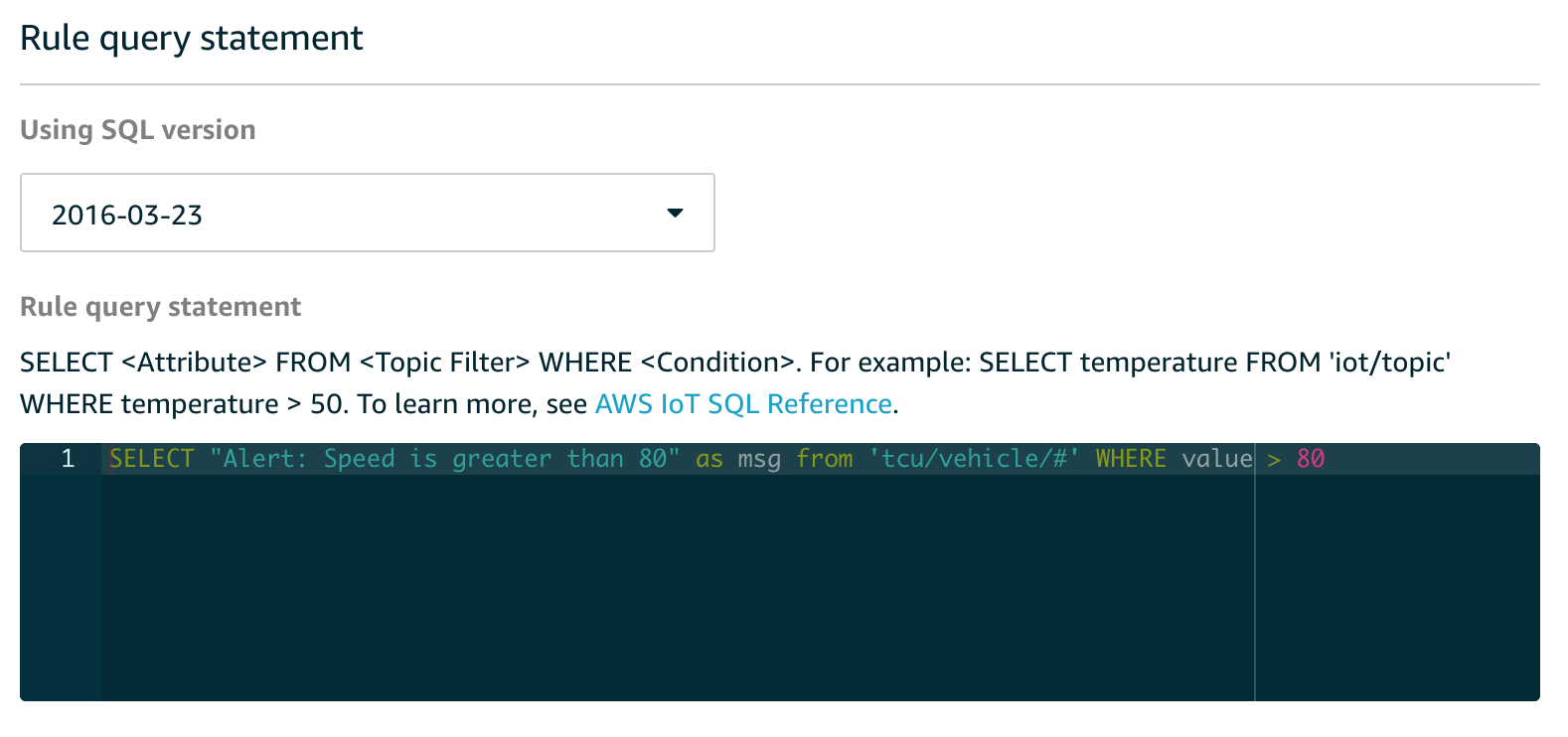
At this point, any messages/alerts coming into our SNS topic also get sent to the SQS queue.

### Step 3 - Create a New IoT Rule

Let’s test what we have so far. We’ll add a rule and action to our AWS IoT setup that will listen to all topics and pass that message onto our SNS topic.

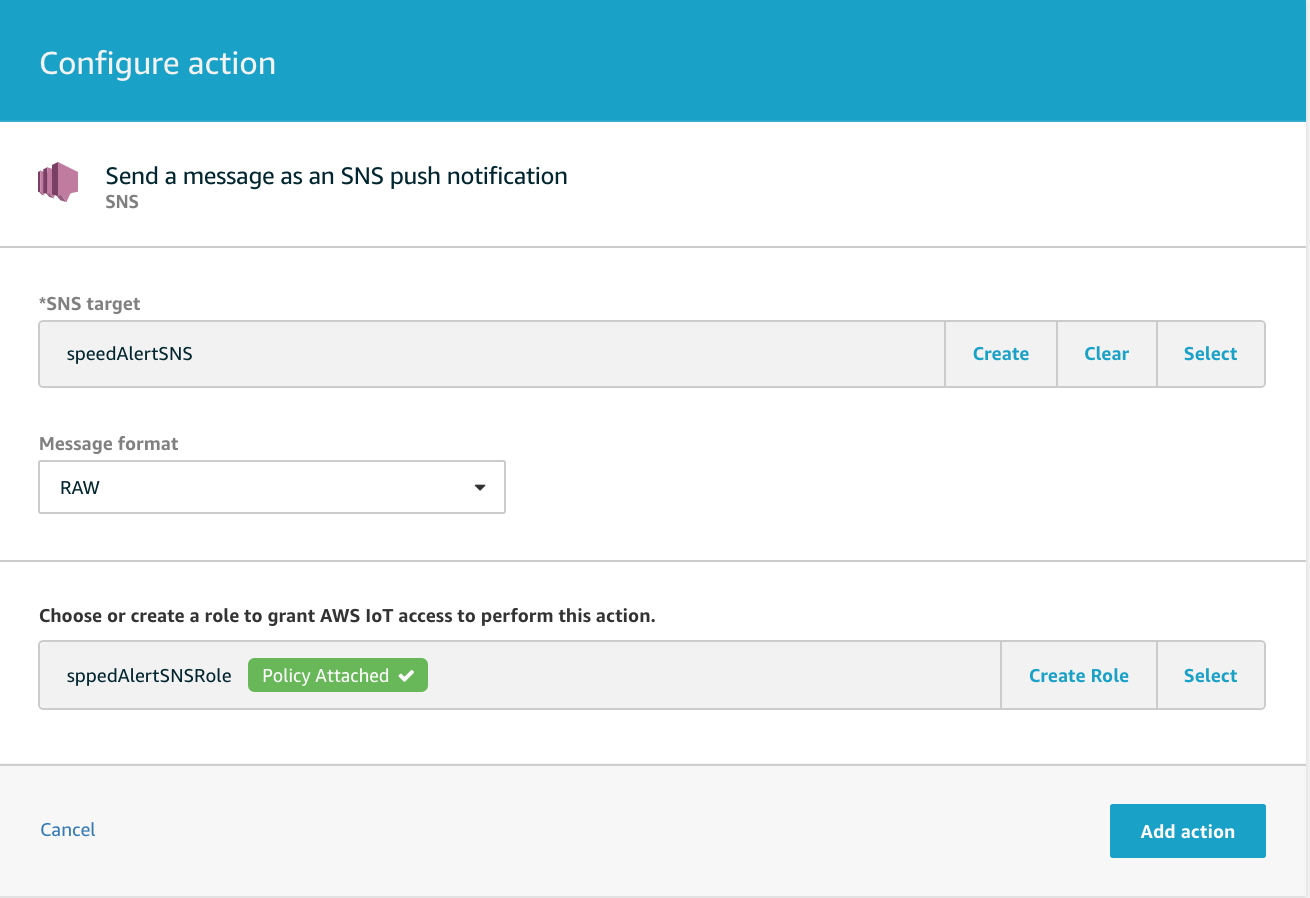
1. Open the AWS IoT dashboard and click **Act**, then **Create Rule**.
2. Create a rule with the following values:
   1. Name: *speedAlertRule*
   2. Rule query statement: SQL statement shown below

SELECT "Alert: Speed is greater than 80" as msg from "tcu/vehicle/#" WHERE value > 80



1. Add Action: **Send a message as an SNS push notification**
2. Choose the **speedAlertSNS** topic we created earlier
3. Message format: **RAW**
4. **Create Role** to grant AWS IoT permissions to perform the action (publish to the speedAlertSNS topic)
5. **Add Action**

**Configure action** page as shown below:



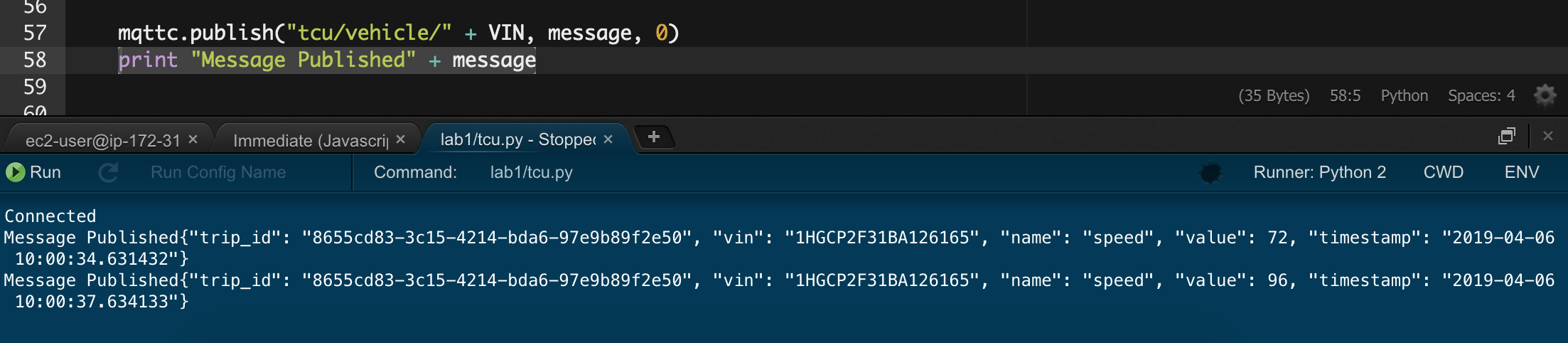
1. Select **Create Rule**.

## Step 4 – Run the tcu and Verify Message

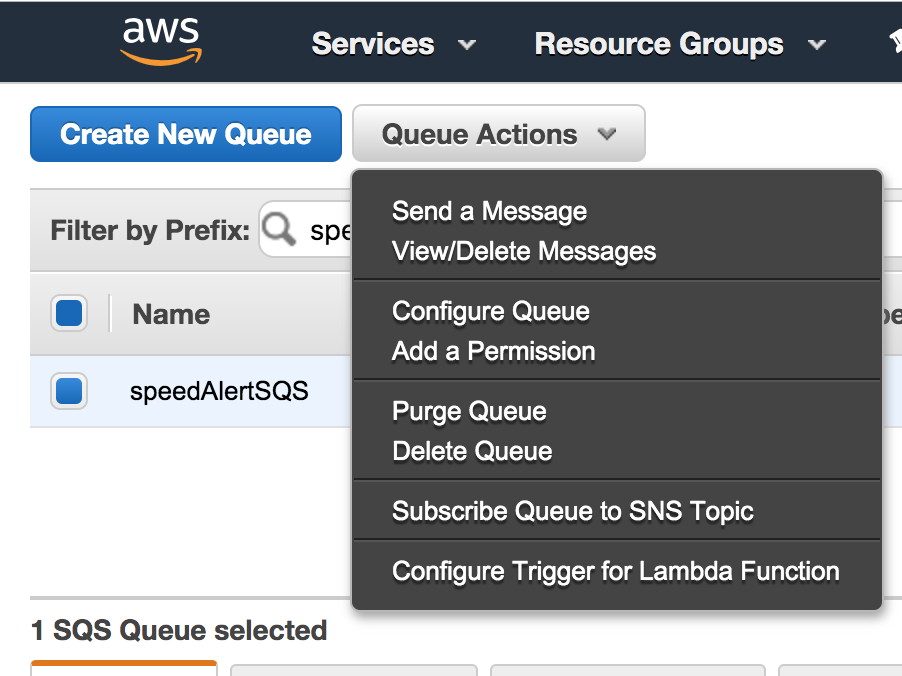
Before we run the tcu simulator, modify the print statement to display the full message on the console. Stop it as soon you find speed value greater than 80 in the cosole output.

print "Message Published" + message

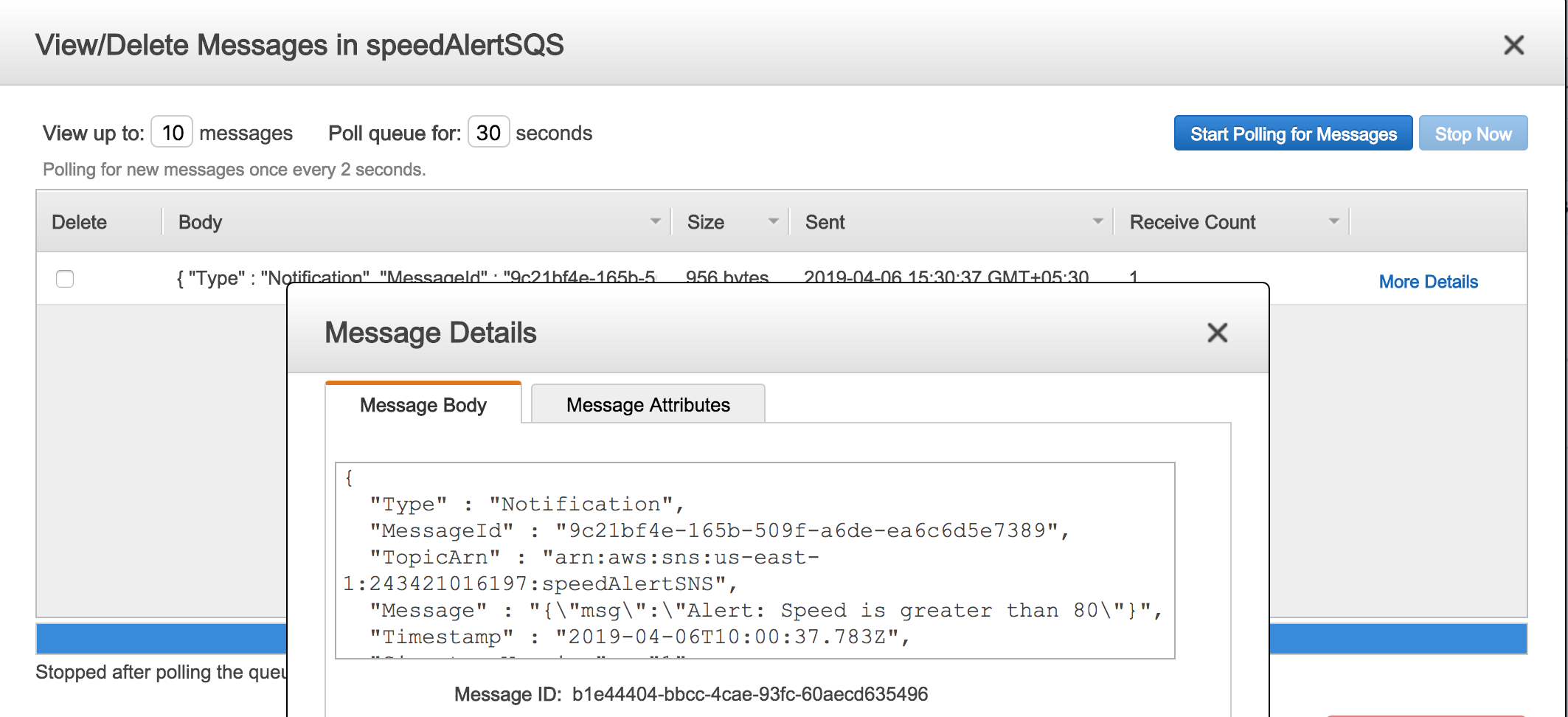
Run the **tcu.py** script in the Cloud9 IDE. You should see something similar to the following:



You should be receiving SMS at your configured mobile phone. If an SMS is not received, then look in the SQS queue. Select queue **speedAlertSQS**, choose **Queue Action,** and then **View/Delete Messages.**



In the **View/Delete Messages** window. Click **Start Polling for Messages**, select the message, and click **More Details**. You should be able to see your speed alert message



You can also look in the SMS CloudWatch logs for failures. To enable Amazon CloudWatch logs for your SMS messages, see: <https://docs.aws.amazon.com/sns/latest/dg/sms_stats_cloudwatch.html>

## Step 5 – Optional Steps

By the end of Lab 3, a simple SMS message is sent to a mobile device. When our TCU reaches 80mph/kmh, the speedAlertRule is configured to send a message to SNS. As an optional step, configure the alert to show the actual speed reached.

Participants can also configure IoT and the simulated vehicle to use a custom endpoint. Configure this from the settings menu in AWS IoT Core. We are using root.CA so you can remove the **-ats** from the custom endpoint name.

# Notices

Customers are responsible for making their own independent assessment of the information in this document. This document: (a) is for informational purposes only, (b) represents current AWS product offerings and practices, which are subject to change without notice, and (c) does not create any commitments or assurances from AWS and its affiliates, suppliers or licensors. AWS products or services are provided “as is” without warranties, representations, or conditions of any kind, whether express or implied. The responsibilities and liabilities of AWS to its customers are controlled by AWS agreements, and this document is not part of, nor does it modify, any agreement between AWS and its customers.

© 2019 Amazon Web Services, Inc. or its affiliates. All rights reserved.